{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"##\*\*Download the dataset\*\*"

],

"metadata": {

"id": "s\_ic6MDnjlML"

}

},

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "Nf3hUzCFO78r",

"outputId": "f01d2311-cac4-4a12-d888-1894730fffb4"

},

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Mounted at /content/drive\n"

]

}

],

"source": [

"from google.colab import drive\n",

"drive.mount('/content/drive')\n"

]

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Import the required library\*\*"

],

"metadata": {

"id": "YKkZVFlTjrra"

}

},

{

"cell\_type": "code",

"source": [

"import pandas as pd\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"import seaborn as sns\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.preprocessing import LabelEncoder\n",

"from tensorflow.keras.models import Model\n",

"from tensorflow.keras.layers import LSTM,Activation, Dense, Dropout, Input,Embedding\n",

"from tensorflow.keras.optimizers import RMSprop\n",

"from tensorflow.keras.preprocessing.text import Tokenizer\n",

"from tensorflow.keras.preprocessing import sequence\n",

"from tensorflow.keras.utils import to\_categorical\n",

"from tensorflow.keras.callbacks import EarlyStopping"

],

"metadata": {

"id": "bZENJh0UPqnM"

},

"execution\_count": 12,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Read the dataset\*\*"

],

"metadata": {

"id": "U0pvoaxYj0qE"

}

},

{

"cell\_type": "code",

"source": [

"import csv\n",

"with open('/content/drive/MyDrive/Colab Notebooks/spam.csv', 'r') as csvfile:\n",

" reader = csv.reader(csvfile)"

],

"metadata": {

"id": "-Nc9b-DYPFK\_"

},

"execution\_count": 8,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df = pd.read\_csv(r'/content/drive/MyDrive/Colab Notebooks/spam.csv',encoding='latin-1')\n",

"df"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 1914

},

"id": "8PWps2LLPpm\_",

"outputId": "a6b4832a-c850-47d8-b30f-796bb9602bc4"

},

"execution\_count": 15,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

" v1 v2 Unnamed: 2 \\\n",

"0 ham Go until jurong point, crazy.. Available only ... NaN \n",

"1 ham Ok lar... Joking wif u oni... NaN \n",

"2 spam Free entry in 2 a wkly comp to win FA Cup fina... NaN \n",

"3 ham U dun say so early hor... U c already then say... NaN \n",

"4 ham Nah I don't think he goes to usf, he lives aro... NaN \n",

"... ... ... ... \n",

"5567 spam This is the 2nd time we have tried 2 contact u... NaN \n",

"5568 ham Will Ì\_ b going to esplanade fr home? NaN \n",

"5569 ham Pity, \* was in mood for that. So...any other s... NaN \n",

"5570 ham The guy did some bitching but I acted like i'd... NaN \n",

"5571 ham Rofl. Its true to its name NaN \n",

"\n",

" Unnamed: 3 Unnamed: 4 \n",

"0 NaN NaN \n",

"1 NaN NaN \n",

"2 NaN NaN \n",

"3 NaN NaN \n",

"4 NaN NaN \n",

"... ... ... \n",

"5567 NaN NaN \n",

"5568 NaN NaN \n",

"5569 NaN NaN \n",

"5570 NaN NaN \n",

"5571 NaN NaN \n",

"\n",

"[5572 rows x 5 columns]"

],

"text/html": [

"\n",

" <div id=\"df-7793bfb6-b702-4827-b115-d6d96cfda229\">\n",

" <div class=\"colab-df-container\">\n",

" <div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>v1</th>\n",

" <th>v2</th>\n",

" <th>Unnamed: 2</th>\n",

" <th>Unnamed: 3</th>\n",

" <th>Unnamed: 4</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>ham</td>\n",

" <td>Go until jurong point, crazy.. Available only ...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>ham</td>\n",

" <td>Ok lar... Joking wif u oni...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>spam</td>\n",

" <td>Free entry in 2 a wkly comp to win FA Cup fina...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>ham</td>\n",

" <td>U dun say so early hor... U c already then say...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>ham</td>\n",

" <td>Nah I don't think he goes to usf, he lives aro...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5567</th>\n",

" <td>spam</td>\n",

" <td>This is the 2nd time we have tried 2 contact u...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5568</th>\n",

" <td>ham</td>\n",

" <td>Will Ì\_ b going to esplanade fr home?</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5569</th>\n",

" <td>ham</td>\n",

" <td>Pity, \* was in mood for that. So...any other s...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5570</th>\n",

" <td>ham</td>\n",

" <td>The guy did some bitching but I acted like i'd...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5571</th>\n",

" <td>ham</td>\n",

" <td>Rofl. Its true to its name</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>5572 rows × 5 columns</p>\n",

"</div>\n",

" <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-7793bfb6-b702-4827-b115-d6d96cfda229')\"\n",

" title=\"Convert this dataframe to an interactive table.\"\n",

" style=\"display:none;\">\n",

" \n",

" <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",

" width=\"24px\">\n",

" <path d=\"M0 0h24v24H0V0z\" fill=\"none\"/>\n",

" <path d=\"M18.56 5.44l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94zm-11 1L8.5 8.5l.94-2.06 2.06-.94-2.06-.94L8.5 2.5l-.94 2.06-2.06.94zm10 10l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94z\"/><path d=\"M17.41 7.96l-1.37-1.37c-.4-.4-.92-.59-1.43-.59-.52 0-1.04.2-1.43.59L10.3 9.45l-7.72 7.72c-.78.78-.78 2.05 0 2.83L4 21.41c.39.39.9.59 1.41.59.51 0 1.02-.2 1.41-.59l7.78-7.78 2.81-2.81c.8-.78.8-2.07 0-2.86zM5.41 20L4 18.59l7.72-7.72 1.47 1.35L5.41 20z\"/>\n",

" </svg>\n",

" </button>\n",

" \n",

" <style>\n",

" .colab-df-container {\n",

" display:flex;\n",

" flex-wrap:wrap;\n",

" gap: 12px;\n",

" }\n",

"\n",

" .colab-df-convert {\n",

" background-color: #E8F0FE;\n",

" border: none;\n",

" border-radius: 50%;\n",

" cursor: pointer;\n",

" display: none;\n",

" fill: #1967D2;\n",

" height: 32px;\n",

" padding: 0 0 0 0;\n",

" width: 32px;\n",

" }\n",

"\n",

" .colab-df-convert:hover {\n",

" background-color: #E2EBFA;\n",

" box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67, 0.15);\n",

" fill: #174EA6;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert {\n",

" background-color: #3B4455;\n",

" fill: #D2E3FC;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert:hover {\n",

" background-color: #434B5C;\n",

" box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",

" filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",

" fill: #FFFFFF;\n",

" }\n",

" </style>\n",

"\n",

" <script>\n",

" const buttonEl =\n",

" document.querySelector('#df-7793bfb6-b702-4827-b115-d6d96cfda229 button.colab-df-convert');\n",

" buttonEl.style.display =\n",

" google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

"\n",

" async function convertToInteractive(key) {\n",

" const element = document.querySelector('#df-7793bfb6-b702-4827-b115-d6d96cfda229');\n",

" const dataTable =\n",

" await google.colab.kernel.invokeFunction('convertToInteractive',\n",

" [key], {});\n",

" if (!dataTable) return;\n",

"\n",

" const docLinkHtml = 'Like what you see? Visit the ' +\n",

" '<a target=\"\_blank\" href=https://colab.research.google.com/notebooks/data\_table.ipynb>data table notebook</a>'\n",

" + ' to learn more about interactive tables.';\n",

" element.innerHTML = '';\n",

" dataTable['output\_type'] = 'display\_data';\n",

" await google.colab.output.renderOutput(dataTable, element);\n",

" const docLink = document.createElement('div');\n",

" docLink.innerHTML = docLinkHtml;\n",

" element.appendChild(docLink);\n",

" }\n",

" </script>\n",

" </div>\n",

" </div>\n",

" "

]

},

"metadata": {},

"execution\_count": 15

}

]

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Pre processing\*\*"

],

"metadata": {

"id": "V8U-LKPMj62i"

}

},

{

"cell\_type": "code",

"source": [

"df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)\n",

"df.info()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "h1OHRZiPQSCd",

"outputId": "88998c13-ca52-4b42-c0a9-a5bad14c1549"

},

"execution\_count": 16,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"<class 'pandas.core.frame.DataFrame'>\n",

"RangeIndex: 5572 entries, 0 to 5571\n",

"Data columns (total 2 columns):\n",

" # Column Non-Null Count Dtype \n",

"--- ------ -------------- ----- \n",

" 0 v1 5572 non-null object\n",

" 1 v2 5572 non-null object\n",

"dtypes: object(2)\n",

"memory usage: 87.2+ KB\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"#####\*\*Count the spam and ham\*\*"

],

"metadata": {

"id": "Mad9SMBLk\_0V"

}

},

{

"cell\_type": "code",

"source": [

"sns.countplot(df.v1)\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 352

},

"id": "oa4TBq1wQtfM",

"outputId": "de262b95-8ad6-4214-e94e-26823d8af913"

},

"execution\_count": 17,

"outputs": [

{

"output\_type": "stream",

"name": "stderr",

"text": [

"/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.\n",

" FutureWarning\n"

]

},

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<matplotlib.axes.\_subplots.AxesSubplot at 0x7f475b28f590>"

]

},

"metadata": {},

"execution\_count": 17

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

},

{

"cell\_type": "code",

"source": [

"X = df.v2\n",

"Y = df.v1\n",

"le = LabelEncoder()\n",

"Y = le.fit\_transform(Y)\n",

"Y = Y.reshape(-1,1)\n",

"X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.20)\n",

"max\_words = 1000\n",

"max\_len = 150\n",

"tok = Tokenizer(num\_words=max\_words)\n",

"tok.fit\_on\_texts(X\_train)\n",

"sequences = tok.texts\_to\_sequences(X\_train)\n",

"sequences\_matrix = sequence.pad\_sequences(sequences,maxlen=max\_len)"

],

"metadata": {

"id": "vrzHnT1pQ8Nf"

},

"execution\_count": 38,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Create Model and add Layers (LSTM,Dense-(Hidden Layers), Output)\*\*"

],

"metadata": {

"id": "IbVdA-GOjJ7Q"

}

},

{

"cell\_type": "code",

"source": [

"def RNN():\n",

" inputs = Input(name='inputs',shape=[max\_len])\n",

" layer = Embedding(max\_words,50,input\_length=max\_len)(inputs)\n",

" layer = LSTM(128)(layer)\n",

" layer = Dense(256,name='FC1')(layer)\n",

" layer = Activation('relu')(layer)\n",

" layer = Dropout(0.5)(layer)\n",

" layer = Dense(1,name='out\_layer')(layer)\n",

" layer = Activation('tanh')(layer)\n",

" model = Model(inputs=inputs,outputs=layer)\n",

" return model\n"

],

"metadata": {

"id": "8KBci7F3jV-O"

},

"execution\_count": 48,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Compile the Model\*\*"

],

"metadata": {

"id": "fayzclNKVDa7"

}

},

{

"cell\_type": "code",

"source": [

"model = RNN()\n",

"model.summary()\n",

"model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy','mse','mae'])"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "-AJAj4T8VB5e",

"outputId": "1984ef38-ecab-4585-93ec-bbd1bd2e180d"

},

"execution\_count": 39,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Model: \"model\_2\"\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

" Layer (type) Output Shape Param # \n",

"=================================================================\n",

" inputs (InputLayer) [(None, 150)] 0 \n",

" \n",

" embedding\_2 (Embedding) (None, 150, 50) 50000 \n",

" \n",

" lstm\_2 (LSTM) (None, 128) 91648 \n",

" \n",

" FC1 (Dense) (None, 256) 33024 \n",

" \n",

" activation\_4 (Activation) (None, 256) 0 \n",

" \n",

" dropout\_2 (Dropout) (None, 256) 0 \n",

" \n",

" out\_layer (Dense) (None, 1) 257 \n",

" \n",

" activation\_5 (Activation) (None, 1) 0 \n",

" \n",

"=================================================================\n",

"Total params: 174,929\n",

"Trainable params: 174,929\n",

"Non-trainable params: 0\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Fit the Model\*\*"

],

"metadata": {

"id": "3y2ZuDwjUZQY"

}

},

{

"cell\_type": "code",

"source": [

"model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10,validation\_split=0.2)\n",

"\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "wFM7XkEaSRi0",

"outputId": "ccd3fad9-1011-44bc-e88d-079ede852d77"

},

"execution\_count": 46,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/10\n",

"28/28 [==============================] - 16s 568ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 2/10\n",

"28/28 [==============================] - 27s 967ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 3/10\n",

"28/28 [==============================] - 15s 536ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 4/10\n",

"28/28 [==============================] - 15s 533ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 5/10\n",

"28/28 [==============================] - 15s 536ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 6/10\n",

"28/28 [==============================] - 15s 532ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 7/10\n",

"28/28 [==============================] - 15s 538ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 8/10\n",

"28/28 [==============================] - 16s 556ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 9/10\n",

"28/28 [==============================] - 16s 570ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n",

"Epoch 10/10\n",

"28/28 [==============================] - 15s 532ms/step - loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val\_loss: 13.2149 - val\_accuracy: 0.1334 - val\_mse: 0.8666 - val\_mae: 0.8666\n"

]

},

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<keras.callbacks.History at 0x7f475002f650>"

]

},

"metadata": {},

"execution\_count": 46

}

]

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Save the Model\*\*"

],

"metadata": {

"id": "TFgUHCLdT9zw"

}

},

{

"cell\_type": "code",

"source": [

"model.save(\"/assignment4\_model.h5\")"

],

"metadata": {

"id": "eYgXFNPjTK7j"

},

"execution\_count": 34,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"##\*\*Test The Model\*\*"

],

"metadata": {

"id": "mhu\_IzLMUMVc"

}

},

{

"cell\_type": "code",

"source": [

"test\_sequences = tok.texts\_to\_sequences(X\_test)\n",

"test\_sequences\_matrix = sequence.pad\_sequences(test\_sequences,maxlen=max\_len)\n",

"accr = model.evaluate(test\_sequences\_matrix,Y\_test)\n",

"print('Test set\\n Loss: {:0.3f}\\n Accuracy:{:0.3f}'.format(accr[0],accr[1]))"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "ZvkJLllpV5EF",

"outputId": "817efbb0-132c-474c-fc61-03ba11e6396a"

},

"execution\_count": 47,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"35/35 [==============================] - 5s 129ms/step - loss: 13.0747 - accuracy: 0.1426 - mse: 0.8575 - mae: 0.8578\n",

"Test set\n",

" Loss: 13.075\n",

" Accuracy:0.143\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"from tensorflow.keras.models import load\_model\n",

"m2 = load\_model(\"/assignment4\_model.h5\")\n",

"m2.evaluate(test\_sequences\_matrix,Y\_test)"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "mcDs1LGyTUzK",

"outputId": "d0f36367-c7bf-44d0-bf5d-0e780054c0ab"

},

"execution\_count": 37,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"35/35 [==============================] - 3s 79ms/step - loss: 0.0989 - accuracy: 0.9865 - mse: 0.0216 - mae: 0.1009\n"

]

},

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"[0.09894020110368729,\n",

" 0.9865471124649048,\n",

" 0.021568873897194862,\n",

" 0.10085303336381912]"

]

},

"metadata": {},

"execution\_count": 37

}

]

}

]

}